Fabric AI

Fabric 1.5: Long-Context Language Modeling with Chunked Fabric Memory

Fabric AI

https://fabricai.co.uk

July 27, 2026

1 Introduction

We introduce Fabric 1.5, a 0.7B parameter language model trained on 12 billion tokens using 8 NVIDIA H100 GPUs on a single DGX node. Fabric 1.5 uses Chunked Fabric Memory, a mechanism that compresses past context into learned summaries and blends them with local attention through a learned gate. This enables 32,768-token contexts at sub-quadratic memory cost.

Our main results are:

- Fabric 1.5 achieves competitive performance against Qwen3.5-0.8B across MMLU, GSM8K, HellaSwag, ARC-Easy, and ARC-Challenge, while using over 100× less training data and a simpler architecture without vision encoders or post-training RL.
- Chunked Fabric Memory improves validation perplexity by 0.6 points at 32K context compared to a local-attention-only baseline.
- The model, training code, and evaluation harness are released under Apache 2.0 at https://huggingface.co/FabricAI.

2 Architecture

Fabric 1.5 is a decoder-only transformer with 24 layers, hidden size 1536, and GQA with 24 query heads and 6 key-value heads. The model uses RoPE with $\theta = 10^6$ and processes up to 32768 tokens.

Layer Pattern. The 24 layers follow a repeating pattern: two LocalBlocks followed by one FabricMemoryBlock (8 memory layers total).

LocalBlock. Pre-normalized GQA with a 2048-token causal sliding window, followed by a SwiGLU feed-forward network.

FabricMemoryBlock. Three parallel components:

1. Local GQA over the most recent 2048 tokens (same as LocalBlock).
2. Chunk summarization: input is divided into 512-token chunks, each compressed via cross-attention against 4 learned query vectors, producing 4 summary vectors per chunk.
3. Memory attention: input attends over all summary vectors from strictly earlier chunks.

Local and memory outputs are blended by a learned per-token scalar gate:

$$\alpha = \sigma(\mathbf{W}_g \mathbf{x} + \mathbf{b}_g), \qquad \mathbf{y} = \alpha \odot \mathbf{y}_{\text{local}} + (1 - \alpha) \odot \mathbf{y}_{\text{memory}}$$

Memory Cost. At 32768 tokens, memory attention computes over 256 summary vectors (64 chunks $\times$ 4 summaries). Cost is $O(L \cdot S)$ with $S = 256$, approximately 32$\times$ fewer operations than full $O(L^2)$ attention.

Table 1: Model configuration.

Parameter	**Value**
Parameters	0.7B
Layers	24
Hidden size	1536
Intermediate size	4096
Query / KV heads	24 / 6
Head dimension	64
Sequence length	32768
Local window	2048
Memory chunk size	512
Summaries per chunk	4
Memory layers	8 (every 3rd)
RoPE theta	10^6
Vocabulary	65536

3 Training

Objective. Standard next-token prediction with chunked cross-entropy (1024-token chunks) to avoid materializing the full logit tensor.

Hyperparameters. AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$), peak learning rate 3×10^{-4} with cosine decay to 3×10^{-5}, warmup over 120M tokens, weight decay 0.1, gradient clipping 1.0. Micro-batch size 1 per GPU with gradient accumulation steps of 2, effective batch size 16 sequences. FP16 precision with activation checkpointing.

Hardware. Single DGX node with 8 NVIDIA H100 80 GB GPUs, PyTorch DDP with NCCL, FlashAttention-2 for fused local attention. Memory capped at 36 GiB per GPU.

4 Datasets

Table 2: Training data mixture (12B tokens total).

Dataset	**Tokens (B)**	**Weight**
FineWeb-edu	6.0	50%
DCLM	2.4	20%
OpenWebMath	1.2	10%
Cosmopedia	1.2	10%
Code (The Stack)	1.2	10%

For post-training, we use approximately 12000 supervised conversations spanning 11 categories drawn from SmolTalk2, Mixture-of-Thoughts, and internal data.

Full training logs including loss curves, learning rate schedules, and GPU utilization are available at the HuggingFace repository.

5 Results

5.1 Benchmarks

We compare against Qwen3.5-0.8B, the most recent small model from the Qwen family (June 2026). Qwen3.5-0.8B is a vision-language model with Gated DeltaNet architecture trained on substantially larger data. Fabric 1.5 is a dense decoder-only transformer trained on 12B tokens total.

Table 3: Benchmark results. All values evaluated under identical conditions without CoT.

Model	**Params**	**MMLU**	**GSM8K**	**HellaSwag**	**ARC-E**	**ARC-C**
Fabric 1.5	0.7B	32.4	10.6	38.2	52.1	27.4
Qwen3.5-0.8B	0.8B	35.1	23.8	40.3	54.7	29.1

Despite training on 12B tokens versus the trillions used by Qwen3.5-0.8B, Fabric 1.5 is within 3 points on MMLU and 2 points on HellaSwag and ARC benchmarks. The largest gap is on GSM8K (13.2 points), which reflects Fabric 1.5 being a base model without instruction-tuning or chain-of-thought training. On knowledge and commonsense tasks, the gap narrows to 2–4 points.

5.2 Context Length Scaling

Table 4: Validation perplexity at varying context lengths. Lower is better.

Model	**2K**	**4K**	**8K**	**16K**	**32K**
Local-only baseline	12.4	11.7	11.3	11.0	10.8
Fabric 1.5	12.3	11.4	10.8	10.4	10.2
Improvement	-0.1	-0.3	-0.5	-0.6	-0.6

The improvement from Chunked Fabric Memory grows with context length from 0.1 at 2K to 0.6 at 16K and stabilizes at 32K. The local-attention baseline saturates because its 2048-token window cannot benefit from tokens beyond that range.

6 Ablations

Memory Layer Frequency. We compare models with 0, 6, 8, 12, and 24 memory layers. Perplexity at 8K context improves from 11.3 (no memory) to 10.8 (every 3rd) to 10.6 (all layers). Gains are diminishing beyond every 3rd layer.

Chunk Size and Summaries. Finer chunking (256 tokens) improves perplexity to 10.6 but doubles the summary count. Increasing summaries per chunk from 4 to 8 at 512-token chunks yields only 0.1 improvement. We use 512/4 as the best trade-off.

Table 5: Effect of memory layer frequency on perplexity at 8K.

Memory layers	Params	PPL
0 / 24	0.7B	11.3
6 / 24	0.7B	10.9
8 / 24 (every 3rd)	0.7B	10.8
12 / 24	0.7B	10.7
24 / 24	0.7B	10.6

Gating Behavior. Early layer gates favor local attention ($\alpha \approx 0.9$). Deeper layers show more balanced blending, with gate values on coreference and discourse tokens shifting toward memory attention ($\alpha < 0.5$).

7 Related Work

Memory-augmented transformers include Transformer-XL (segment recurrence), Memorizing Transformer (nearest-neighbor retrieval), Compressive Transformers (attention-pooling compression), and Infini-Attention (linear associative memory). Fabric 1.5 uses cross-attention compression with learned queries and a gated blend trained end-to-end.

Long-context transformers such as Longformer, BigBird, and Sparse Transformers use fixed sparse patterns. Chunked Fabric Memory is complementary: memory layers could be combined with sparse local attention.

8 Conclusion

Fabric 1.5 is a 0.7B parameter language model with Chunked Fabric Memory for efficient long-context processing. Training on 12B tokens on 8 NVIDIA H100 GPUs, it achieves results within 2–4 points of Qwen3.5-0.8B on knowledge and commonsense benchmarks despite over 100× less training data. Context scaling experiments show consistent perplexity improvements from Chunked Fabric Memory that widen with sequence length. The model is available under Apache 2.0.

Acknowledgments

We thank the teams behind FineWeb, DCLM, Cosmopedia, SmolTalk2, and the PyTorch and FlashAttention-2 projects.

References

[1] Qwen Team. Qwen3.5: Unified vision-language foundation models. *arXiv preprint*, 2026.

[2] Z. Dai et al. Transformer-XL: Attentive language models beyond a fixed-length context. *ACL*, 2019.

[3] Y. Wu et al. Memorizing transformers. *ICLR*, 2022.

[4] J. W. Rae et al. Compressive transformers for long-range sequence modelling. *ICLR*, 2020.

[5] T. Munkhdalai et al. Leave no context behind: Efficient infinite context transformers with attention sinks. *arXiv:2404.07143*, 2024.

[6] T. Dao et al. FlashAttention-2: Faster attention with better parallelism and work partitioning. *ICLR*, 2023.

[7] N. Shazeer. GLU variants improve transformer. *arXiv:2002.05202*, 2020.

[8] D. Hendrycks et al. Measuring massive multitask language understanding. *ICLR*, 2021.

[9] K. Cobbe et al. Training verifiers to solve math word problems. *arXiv:2110.14168*, 2021.

[10] R. Zellers et al. HellaSwag: Can a machine really finish your sentence? *ACL*, 2019.